

PRELIMINARY
FOR NASA & LPT PROGRAM PARTICIPANTS INTERNAL USE
ONLY

NASA Grant NAG 2099

**Large Eddy Simulation of Flow in Turbine Cascades
Using LESTool and UNCLE Codes**

Final Progress Report

P.G. Huang

Department of Mechanical Engineering

University of Kentucky

Lexington, Kentucky 40506-0503

Submitted to:

Dr. David Ashpis, Technical Monitor

NASA Glenn Research Center

Cleveland, OH 44135

August 31, 2004

**Large Eddy Simulation of Flow in Turbine Cascades
Using LESTool and UNCLE Codes**

P.G. Huang

Department of Mechanical Engineering

University of Kentucky

Lexington, Kentucky 40506-0503

Progress Summary

During the period December 23, 1997 and December August 31, 2004, we accomplished the development of 2 CFD codes for DNS/LES/RANS simulation of turbine cascade flows, namely LESTool and UNCLE. LESTool is a structured code making use of 5th order upwind differencing scheme and UNCLE is a second-order-accuracy unstructured

code. LESTool has both Dynamic SGS and Sparlart's DES models and UNCLE makes use of URANS and DES models. The current report provides a description of methodologies used in the codes.

1. Introduction

Flow transition plays an important role in turbomachinery applications. The majority of boundary layer flows in turbomachines involve flow transition under the effects of freestream turbulence, diverse pressure gradients, wide range of Reynolds numbers, flow separation, and unsteady wake-boundary layer interactions.

Prediction of this type of complex flows is an important element in analysis and performance evaluation of gas turbine engine components and ultimately in the design of more efficient jet engines. Especially, in low pressure turbine applications prediction of transition becomes pivotal in terms of efficiency. For low pressure turbines the flow is mostly turbulent at the high Reynolds number conditions encountered at take off and the efficiency is at its design maximum. However, at high altitudes and cruise speeds which correspond to lower Reynolds number conditions, unpredicted losses and substantial drops in efficiency have been observed. These losses are attributed to flow separation on the suction surface of the turbine blades. At low Reynolds numbers, the boundary layers on the airfoil surface have a tendency to remain laminar and hence the flow may separate before it becomes turbulent, causing increase in fuel consumption and drop in efficiency. The impact of such losses is directly felt on the operation costs. It has been estimated that a 1% improvement in the efficiency of a low pressure turbine would result in a saving of \$52,000 per year on a typical airliner.

In order to calculate the losses and heat transfer on various components of gas turbine engines, and to be able to improve component efficiencies and reduce losses through better designs, accurate prediction of transitional boundary layers is essential. When one deals with a complex fluid phenomena like a transition, separation and turbulence, several hundred millions grid points are needed to resolve boundary layers and other flow structures correctly. We have started to develop technology to make such large scale simulations not only possible at supercomputing centers like NCSA or NAS but on inexpensive, high-performance clusters of PCs, or "Beowulfs". These clusters are specialized for CFD applications, using the novel approach that the hardware, operating system, and application code are optimized together rather than separately. A Honorable

Mention in the Price/Performance Category of the Gordon Bell Prize was awarded for this approach at IEEE/ACM SC2000 Conference on High-Performance Networking and Computing.

Several turbulence test cases have been computed and an overview of the results is given.

2. Code Descriptions

- (1) A description of LESTool is give in Appendix 1.
- (2) A description of UNCLE is given in Appendix 2.

3. Publications by the PI

- (1) Th. Hauser, T. Mattox, R. LeBeau, H. Dietz and P. Huang. Code optimizations for complex microprocessors applied to CFD software. *SIAM Journal for Scientific Computing (accepted for publication)*, 2003.
- (2) Th. Hauser, R. LeBeau, T. Mattox, P. Huang and H. Dietz. Improving the performance of Computational Fluid Dynamics codes on Linux Cluster Architectures. 16th AIAA Computational Fluid Dynamics Conference. Orlando, Florida, June 23-26, 2003. AIAA.
- (3) Th. Hauser, T. Mattox, R. LeBeau, H. Dietz and P. Huang. Scrutinizing CFD performance on multiple Linux cluster architectures. Presentation at the Clusterworld Conference & Expo, June 23-26, San Jose, California, 2003.
- (4) Th. Hauser, T. Mattox, R. LeBeau, H. Dietz and P. Huang. A comparative study of the performance of a CFD program across different Linux cluster architectures. Proceedings of the third LCI international confernece on linux clusters. St. Petersburg, FL, 2002.
- (5) Hauser, T., R. LeBeau, T. Mattox, H. Dietz and P. Huang. High-Cost CFD on a Low-Cost Cluster. Proceedings of 8th National CFD Conference. 2001. Invited Keynote Talk, E-land Taiwan, Aug 18-20.

(6) Th. Hauser, T. Mattox, R. LeBeau, H. Dietz and P. Huang. High-Cost CFD on a Low-cost Cluster. Regular paper at SC2000 and honorable mention for the Gordon Bell award in the category "price-performance". 2000.

(7) Th. Hauser and P. Huang. Numerical simulation of turbulent spots. 25th Annual Dayton-Cincinnati Aerospace Science Symposium, 2000.

(8) Th. Hauser and P. Huang. A hierarchical parallelization concept for a high-performance Navier-Stokes solver. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99). 1999.

(9) Th. Hauser and P. Huang. Large eddy simulation of low pressure turbine flow. 24th Annual Dayton-Cincinnati Aerospace Science Symposium, 1999.

(10) Th. Hauser and P. Huang. Shared Memory Parallelization of an implicit ADI-type CFD code. In C. Lin and et. al, editors, Parallel computational fluid dynamics, development and applications of parallel technology, proceedings of the Parallel CFD'98 Conference, pages 145-152. 1999. Elsevier Science B.B.

(11) R. Savaram, T. Hauser and P. Huang. DNS and LES of homogeneous turbulence. 24th Annual Dayton-Cincinnati Aerospace Science Symposium, 1999.

(12) Th. Hauser and P. Huang. Shared Memory Parallelization of an implicit ADI-type CFD code. Technical report, NASA-CR-208688, NASA, 1998.

4. Award received by the PI

A Honorable Mention in the Price/Performance Category of the Gordon Bell Prize was awarded for this approach at IEEE/ACM SC2000 Conference on High-Performance Networking and Computing \cite{hauser2000}.

Appendix I - LESTool

LESTool: A CFD Program for 3D unsteady flows

Thomas Hauser

George Huang

30th January 2004

Chapter 1

Introduction

Flow transition plays an important role in turbomachinery applications. The majority of boundary layer flows in turbomachines involve flow transition under the effects of freestream turbulence, diverse pressure gradients, wide range of Reynolds numbers, flow separation, and unsteady wake-boundary layer interactions.

Prediction of this type of complex flows is an important element in analysis and performance evaluation of gas turbine engine components and ultimately in the design of more efficient jet engines. Especially, in low pressure turbine applications prediction of transition becomes pivotal in terms of efficiency. For low pressure turbines the flow is mostly turbulent at the high Reynolds number conditions encountered at take off and the efficiency is at its design maximum. However, at high altitudes and cruise speeds which correspond to lower Reynolds number conditions, unpredicted losses and substantial drops in efficiency have been observed ([?], [?], [?]). These losses are attributed to flow separation on the suction surface of the turbine blades. At low Reynolds numbers, the boundary layers on the airfoil surface have a tendency to remain laminar and hence the flow may separate before it becomes turbulent, causing increase in fuel consumption and drop in efficiency. The impact of such losses is directly felt on the operation costs. It has been estimated that a 1% improvement in the efficiency of a low pressure turbine would result in a saving of \$52,000 per year on a typical airliner ([?]).

In order to calculate the losses and heat transfer on various components of gas turbine engines, and to be able to improve component efficiencies and reduce losses through better designs, accurate prediction of transitional boundary layers is essential ([?]). When one deals with a complex fluid phenomena like a transition, separation and turbulence, several hundred millions grid points are needed to resolve boundary layers and other flow structures correctly. We have started to develop technology to make such large scale simulations not only possible at su-

percomputing centers like NCSA or NAS but on inexpensive, high-performance clusters of PCs, or "Beowulfs" [?]. These clusters are specialized for CFD applications, using the novel approach that the hardware, operating system, and application code are optimized together rather than separately. A Honorable Mention in the Price/Performance Category of the Gordon Bell Prize was awarded for this approach at IEEE/ACM SC2000 Conference on High-Performance Networking and Computing [?].

Several turbulence testcases have been computed and an overview of the results is given.

Chapter 2

Turbulence Models

2.1 LES

Filtered equations

Large-eddy simulation (LES) is based on the definition of a filtering operation: a filtered (or large-scale, or resolved) quantity, denoted by an overbar, is defined as

$$\bar{f}(\mathbf{x}) = \int_D f(\mathbf{x}') G(\mathbf{x}, \mathbf{x}') d\mathbf{x}', \quad (2.1)$$

where D is the entire domain and G is the filter function, which determines the size and structure of the small scales. For compressible flow the large-scale equations are operationally simpler if Favre filtered quantities are used. A Favre filtered variable is defined as $\tilde{f} = \bar{\rho f} / \bar{\rho}$. Applying the spatial filter G to the governing equations leads to

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{u}_j) = 0, \quad (2.2)$$

$$\frac{\partial \bar{\rho} \tilde{u}_i}{\partial t} + \frac{\partial}{\partial x_j} (\bar{\rho} \tilde{u}_i \tilde{u}_j) = - \frac{\partial \bar{p}}{\partial x_i} + \frac{\partial \tilde{\tau}_{ji}}{\partial x_j} - \underbrace{\frac{\partial \tau_{ji}^{SGS}}{\partial x_j}}_I + \underbrace{\frac{\partial}{\partial x_j} (\bar{\tau}_{ji} - \tilde{\tau}_{ji})}_{II} \quad (2.3)$$

$$\begin{aligned} \frac{\partial \bar{\rho} \tilde{E}}{\partial t} + \frac{\partial}{\partial x_j} ((\bar{p} \tilde{E} + \bar{p}) \tilde{u}_j) &= - \frac{\partial \tilde{q}_i}{\partial x_j} + \frac{\partial}{\partial x_j} (\tilde{\tau}_{ji} \tilde{u}_i) \\ &\quad - \underbrace{c_p \frac{\partial}{\partial x_j} (q_j^{SGS})}_{III} - \frac{1}{2} \frac{\partial}{\partial x_j} (\tau_{ii}^{SGS} u_j) - \underbrace{\frac{\partial}{\partial x_j} D_j}_{IV} \end{aligned} \quad (2.4)$$

4. This term is similar to the heat flux and can be modelled accordingly
5. This term is neglected because of the same reasons as 2
6. This term is the viscous SGS work and is neglected

To summarize the following quantities will be modeled in our implementation:

1. SGS stresses τ_{ij}^{SGS}
2. SGS heat flux q_j^{SGS}

2.1.1 Smagorinsky Turbulence Model

2.1.2 Dynamic SGS model

For the left-hand side a trace-free Smagorinsky eddy viscosity model is used for τ_{ij}^{SGS} . However the eddy viscosity coefficient will be a function of the instantaneous flow variables. Let

$$\tau_{ij}^{SGS} - \frac{1}{3} q^2 \delta_{ij} = -2C \bar{\rho} \Delta^2 |\tilde{S}| \tilde{S}_{ij}^* \quad (2.12)$$

where $\tilde{S}_{ij}^*$ is the trace-free rate of the strain tensor

$$\tilde{S}_{ij}^* = \tilde{S}_{ij} - \frac{\delta_{ij}}{3} \tilde{S}_{kk} \quad (2.13)$$

and $|\tilde{S}|$ is the norm of $\tilde{S}_{ij}$

$$|\tilde{S}| = \sqrt{2 \tilde{S}_{ij} \tilde{S}_{ij}} \quad (2.14)$$

The isotropic part of the SGS Reynolds stress tensor $q^2 = \tau_{kk}$ has to be modeled separately. It is parametrized using Yoshizawa's [?], [?] expression:

$$q^2 = 2C_\epsilon \bar{\rho} \Delta^2 |\tilde{S}|^2 \quad (2.15)$$

To compute C_ϵ , the trace of eq. ?? together with 2.12 is used:

$$\overline{\rho \tilde{u}_i \tilde{u}_i} - \frac{\overline{\rho \tilde{u}_i \tilde{u}_i}}{\bar{\rho}} = 2C_\epsilon \left(\overline{\hat{\rho} \hat{\Delta}^2 |\tilde{S}|^2} - \Delta^2 \overline{\hat{\rho} |\tilde{S}|^2} \right) \quad (2.16)$$

Germano et. al. [?] showed that expressions similar to the ones multiplying C_ϵ can become zero. Therefore an averaging procedure is needed to make the determination of the SGS coefficients well conditioned. It is assumed that C_ϵ is

independent of the directions in which the flow is homogeneous. Volume averaging leads to

$$C_i \Delta^2 = \frac{\langle L_{kk} \rangle}{2 \langle \hat{\rho} \left(\frac{\hat{\Delta}}{\Delta} \right)^2 |\hat{S}|^2 - \hat{\rho} |\hat{S}|^2 \rangle} \quad (2.17)$$

To obtain C we also use the model of eq. 2.12 for the test field stresses

$$T_{ij} - \frac{1}{3} T_{kk} \delta_{ij} = -2C \hat{\rho} \hat{\Delta}^2 |\hat{S}| \hat{S}_{ij}^* \quad (2.18)$$

This leads to

$$L_{ij} = \frac{1}{3} L_{kk} \delta_{ij} - 2C \hat{\rho} \hat{\Delta}^2 |\hat{S}| \hat{S}_{ij}^* + 2C \Delta^2 \hat{\rho} |\hat{S}| \hat{S}_{ij}^* \quad (2.19)$$

$$= \frac{1}{3} L_{kk} \delta_{ij} + 2C \Delta^2 M_{ij} \quad (2.20)$$

with M_{ij}

$$M_{ij} = \hat{\rho} |\hat{S}| \hat{S}_{ij}^* - \hat{\rho} \left(\frac{\hat{\Delta}}{\Delta} \right)^2 |\hat{S}| \hat{S}_{ij}^* \quad (2.21)$$

Contracting eq. ?? with M_{ij} which was recommended by Lilly is outlined below. Since eq. ?? represents six independent equations in one unknown, its error can be minimized by applying a least squares approach. Define G as

$$G = (L_{ij} - \frac{1}{3} \delta_{ij} L_{kk} - 2C M_{ij})^2 \quad (2.22)$$

By setting $\partial G / \partial C = 0$, C is evaluated as

$$C \Delta^2 = \frac{1}{2} \frac{\langle L_{ij} M_{ij} - \frac{1}{3} L_{kk} M_{kk} \rangle}{\langle M_{ij} M_{ij} \rangle} \quad (2.23)$$

where $\langle \rangle$ denotes a spatial averaging operation.

2.2 DES model

The DES turbulence model is based on the blending of RANS and LES to provide a model that can accurately predict unsteady flows without requiring high-precision grids near the walls. A RANS turbulence model is used to simulate the turbulence within the boundary layer, gradually subsuming into the LES model as one moves

away from the wall. Further details on this technique can be found in [?], [?] and others. The RANS turbulence model used in LESTool is the one-equation Spalart-Allmaras model,

$$\frac{D\tilde{\nu}}{Dt} = P_k - L + \frac{1}{\sigma} \{ \nabla \cdot [(\nu + \tilde{\nu}) \nabla \tilde{\nu}] \}, \quad (2.24)$$

where

$$P_k = c_{b1} \left| f_{v2} \Omega + f_{v2} \frac{\tilde{\nu}}{\kappa^2 d^2} \right| \tilde{\nu} + \frac{1}{\sigma} c_{b2} (\nabla \tilde{\nu} \cdot \nabla \tilde{\nu}), \quad L = c_{w1} f_w \left(\frac{\tilde{\nu}}{d} \right)^2, \quad (2.25)$$

$$f_{v1} = \frac{\chi^6}{\chi^6 + c_{v1}}, \quad f_{v2} = \left(1 + \frac{\chi}{c_{v2}} \right)^{-6}, \quad f_{v3} = \frac{1}{\chi} (1 - f_{v2}) (1 + f_{v1} \chi), \quad \chi \equiv \frac{\tilde{\nu}}{\nu}, \quad (2.26)$$

$$f_w = g \left[\frac{1 + c_{w2}}{g^6 + c_{w2}^6} \right]^{\frac{1}{6}}, \quad g = r + c_{w2} (r^6 - r), \quad r \equiv \frac{\tilde{\nu}}{\tilde{S} \kappa^2 d^2}, \quad \tilde{S} = \left| f_{v3} \Omega + f_{v2} \frac{\tilde{\nu}}{\kappa^2 d^2} \right|, \quad (2.27)$$

with the coefficients given below

$$c_{b1} = 0.1355, \quad c_{b2} = 0.622, \quad c_{v1} = 7.1, \quad c_{v2} = 5.0, \quad c_{w1} = c_{b1}/\kappa^2 + (1 + c_{b2})/\sigma,$$

$$c_{w2} = 0.3, \quad c_{w3} = 2, \quad \sigma = 2/3, \quad \kappa = 0.41, \quad \Omega = |\nabla \times \vec{u}|,$$

and d is the distance to the nearest wall. Note that this formulation does not include the trip functions. Large values of r can be truncated at 10 or so. The turbulent viscosity for the momentum equations is found from $\nu_t = \tilde{\nu} f_{v1}$. For the DES approach the length scale d is changed into

$$d^* = \min(d, \max(\Delta x, \Delta y, \Delta z) C_i) \quad (2.28)$$

with $C_i = 0.65$. This gives the turbulence model behaviour near the wall and LES type behavior away from the wall

$$(D^L + \Delta t L_y^L) d(\delta U^2) = D^L [(\delta U^1)^m - (\delta U^2)^{m-1}] \quad (3.5)$$

$$(D^L + \Delta t L_x^L) d(\delta U^3) = D^L [(\delta U^2)^m - (\delta U^3)^{m-1}] \quad (3.6)$$

This algorithm is easily parallelizable because we loop through a three dimensional array plane by plane. The outline of the algorithm is the following:

1. for each k: solve equations (2) and (3) in independent i-j planes
2. for each j: solve equations (4) and (2) in independent k-i planes
3. for each i: solve equations (3) and (4) in independent j-k planes

This means after completion of the above algorithm we have already completed two iterations. From our experience three to four iterations are needed for each time-step to reach the required accuracy.

Fortran 90 was chosen as the programming language for this project because of the new powerful array syntax, the addition of derived types and the module concept. The code has been developed based on an object-oriented methodology to simplify the transition from the Cartesian implementation to an implementation for general coordinates and chimera-type overlapping grids. The new features of Fortran 90 like abstract data types or generic programming support an object-oriented programming style even though it is not designed to be an object-oriented language.

Chapter 3

Numerical method

The compressible Navier-Stokes equations are discretized using an iterative diagonal dominant ADI (DD-ADI) algorithm that can be written in the following form:

$$\left[D^L + \Delta t L_x^L + \Delta t L_y^L + \Delta t L_z^L \right] \delta U = \Delta RHS^H \quad (3.1)$$

The right-hand side is approximated by the newly develop ENO-Pade method [?] for the convective flux and sixth-order central for the diffusive terms. The basic algorithm is the following. First the flux at each cell face is computed as shown below

$$F_{i+1/2} = F(Q_{i+1/2}^H) - \frac{1}{2} |A_{i+1/2}^H| (Q_{i+1/2}^H - Q_{i+1/2}^L) \quad (3.2)$$

Quantities with the superscript H are computed using a 8th-order central interpolation method. For the quantities with the superscript R or L the standard ENO-interpolation [?]. Then the flux is differentiated using a cell centered Pade formula [?]:

$$\alpha \phi'_{i-1} + \phi'_i + \alpha \phi'_{i+1} = a \frac{\phi_{i+1/2} - \phi_{i-1/2}}{h} + b \frac{\phi_{i+3/2} - \phi_{i-3/2}}{3h} \quad (3.3)$$

The left-hand side is approximated by a lower-order method. The high order solution can be achieved by performing inner iterations since the order of accuracy is not affected by low order treatment of the left-hand side.

The proposed algorithm has the following form:

$$(D^L + \Delta t L_y^L) \delta (U^1) = D^L \left[(\delta U^0)^{n-1} - (\delta U^1)^{n-1} \right] \quad (3.4)$$

Chapter 4

Parallelization

4.1 Shared Memory parallel approach

The LESTool code was parallelized in a straightforward manner: The code was compiled with the `-mp` option to enable parallelism in the compile phase. This was combined with placing OpenMP compiler directives, which instruct the compiler to generate code that will execute in parallel.

These directives were placed at key spots within the code for the greatest parallel efficiency. This resulted in a decomposing of the 3D problem into groups of 1D lines, with each group assigned to a dedicated processor.

The efficiency of the parallel decomposition was enhanced by the use of the first touch policy which is specific to the SGI Origin 2000. This means that memory allocated is physically placed on the processor which touches the memory location first. This means that all large three dimensional blocks of memory were initialized in planes of constant k to get a memory distribution which is favorable for our algorithmic design (see figure 4.1).

Since we target shared memory computers there is no explicit data redistribution needed. Instead we split up the k -sweeps among the processors by partitioning work along the y -axis (fig. 4.2). This is much simpler and easier to implement compared to the distributed memory concept.

4.2 Distributed memory approach

In order to achieve better portability and performance on a wider range of computational platforms a distributed approach was also implemented. The parallel structure of the distributed version of LESTool is based on splitting the computational grid into sub-blocks, which are then distributed to each processor (figure

Chapter 4

Parallelization

4.1 Shared Memory parallel approach

The LESTool code was parallelized in a straightforward manner: The code was compiled with the `-mp` option to enable parallelism in the compile phase. This was combined with placing OpenMP compiler directives, which instruct the compiler to generate code that will execute in parallel.

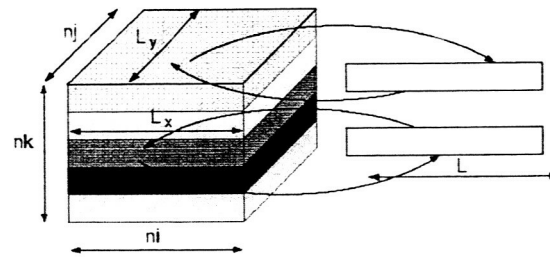
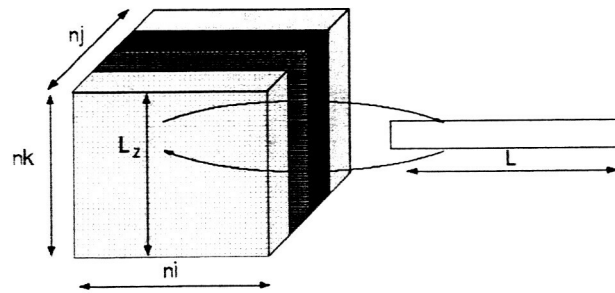
These directives were placed at key spots within the code for the greatest parallel efficiency. This resulted in a decomposing of the 3D problem into groups of 1D lines, with each group assigned to a dedicated processor.

The efficiency of the parallel decomposition was enhanced by the use of the first touch policy which is specific to the SGI Origin 2000. This means that memory allocated is physically placed on the processor which touches the memory location first. This means that all large three dimensional blocks of memory were initialized in planes of constant k to get a memory distribution which is favorable for our algorithmic design (see figure 4.1).

Since we target shared memory computers there is no explicit data redistribution needed. Instead we split up the k -sweeps among the processors by partitioning work along the y -axis (fig. 4.2). This is much simpler and easier to implement compared to the distributed memory concept.

4.2 Distributed memory approach

In order to achieve better portability and performance on a wider range of computational platforms a distributed approach was also implemented. The parallel structure of the distributed version of LESTool is based on splitting the computational grid into sub-blocks, which are then distributed to each processor (figure

Figure 4.1: Partitioning of the data along the k -axisFigure 4.2: Sweep along the zz -planes

4.3). For a complex geometry, this is a nontrivial exercise where an uneven load balance across different processors could significantly reduce the computational efficiency of the overall code. The partitioning of the grid into sub-blocks is performed independently of the grid generation, using virtual partitions to define the domains corresponding to each processor. The current algorithm governing the partition process is based on spectral recursive bisection in conjunction with a small database of the computational speeds associated with each node. The splitting is performed as a preprocessing task, generating a file that maps the grid sub-blocks onto the processors. This file is the only difference between performing a serial computation and a parallel computation on a distributed machine.

Communications between the grid sub-blocks occurs when the sub-blocks exchange data about the flow variables at the boundaries. As shown in figure 4.3, the flow variables on the edge of one grid block are communicated to the dummy points of the neighboring grid block, and vice versa. LESTool requires such a communication step after each update, or subiteration, of the flow variables. The low-level implementation of the communication between the sub-blocks uses a MPI-based communications system. The communication model is a mailbox algorithm where the data is sent in a non-blocking communication mode as early as possible.

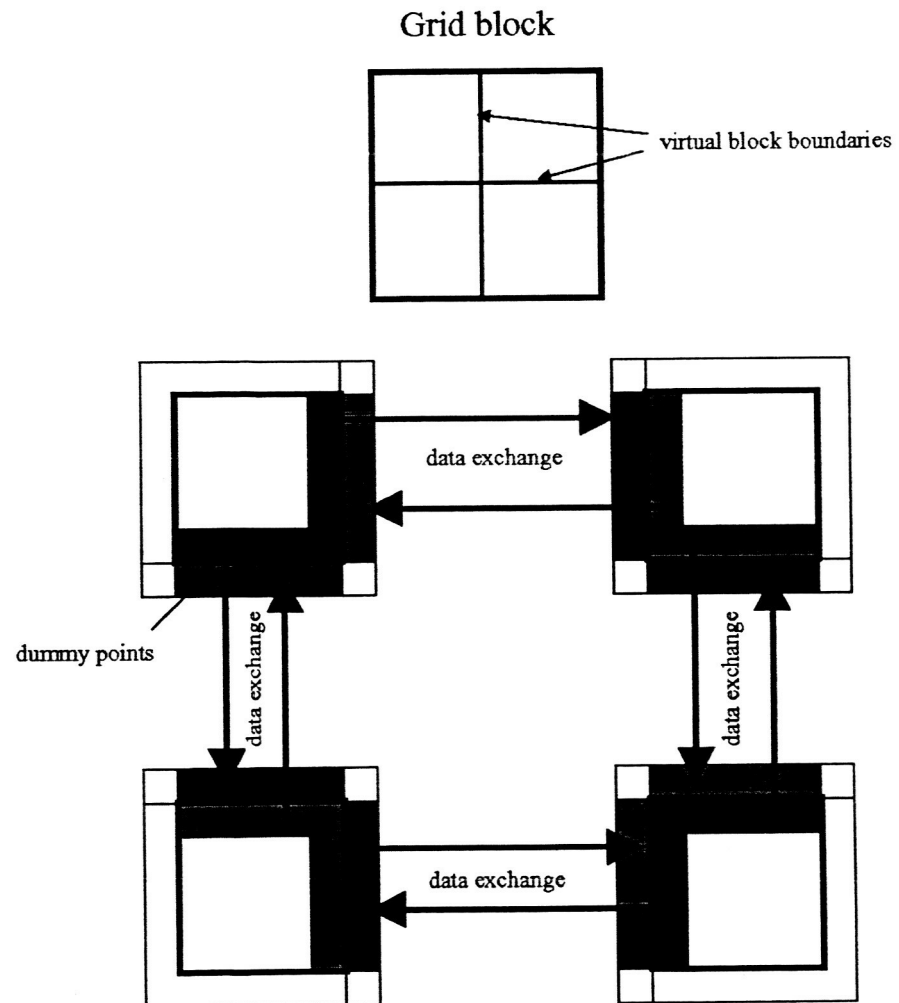


Figure 4.3: Communication pattern for the distributed approach

Chapter 5

Accuracy and Performance of LESTool

5.1 Accuracy of the ENO-Pade method

In a previous proposal a fifth-order upwind scheme has been proposed to simulate flow around a low pressure turbine. This method showed too much numerical damping so that the transition and separation on the low-pressure turbine couldn't be simulated. In order to develop a code suitable for transition and turbulence a new numerical method - the ENOPade method - was developed [?]. This method combines the stability of the ENO scheme with the high-order accuracy of the Pade method. Several testcases have been computed and the damping of the new scheme is much lower as you can see below from one test case. This case consists of the unsteady advection of a vortex in a uniform flow. Here the capabilities of the different numerical schemes to accurately advect the vortex structures are tested which is critical to LES/DNS simulations.

5.2 Performance Results

5.2.1 Single processor performance tuning

The key point on cache based architectures to achieve high floating point performance is cache-awareness of the algorithm. This stage of tuning is very important because efficient utilization of the cache architecture assures good overall performance. Single processor performance is the basic step to get good overall performance for the parallel computation.

During the development of the code special care was taken by placing variables

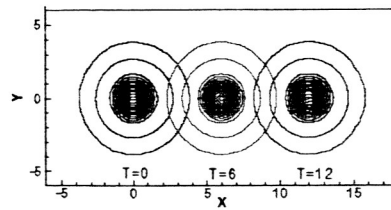


Figure 5.1: Contour lines of vorticity magnitude at three time intervals of ENOPade method

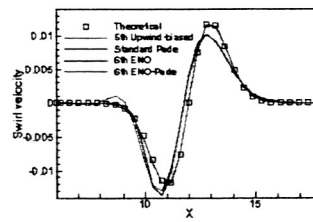


Figure 5.2: Swirl velocities of the vortex along $y=0$ at $T = 12$

which are used concurrently in the calculation close together in the main memory. One major detail is that the first index in the data arrays is used for addressing the different components of global variables, such as conservative variables. This is contrary to a vector architecture, where the index for a variable is normally addressed by the last index in an array. The data is copied into one-dimensional scratch arrays which fit into the second level cache. The main algorithm is performed on these scratch arrays which have a memory layout and access pattern (stride one) optimal for the cache architecture. This measure alone resulted in a floating-point performance of 60 MFLOPS.

To achieve further speedup of the code, the main bottle necks identified were the tridiagonal and block-tridiagonal matrix solvers. To increase the utilization of memory bandwidth, all arithmetic operations involving the 5×5 block matrices, used in block tridiagonal and periodic block tridiagonal solvers, were unrolled. This resulted in a much longer and less desirable code, but the performance results outweigh this disadvantage. The LESTool code achieves a floating-point performance of about 120 MFLOP/s on a 180 MHz Origin 2000. We consider this a very satisfactory performance.

Single processor performance

Single processor run times for a single test case sufficiently large so that it does not fit in the L2 cache of all our CPUs has been chosen. The runtime of the code / time step is reported in table 5.1. The LosLobos results (733 MHz Pentium III) are much worse than the Athlon results - this is probably related to (2 computations/2 data transfers per cycle). However, we are not absolutely confident that this is the optimal possible performance of LESTool on LosLobos, and as such subsequent results will emphasize the KLAT2 and KFC1 outcomes.

Table 5.1: Execution time in seconds on different Linux platforms for a single time step for the 64° test case

	Compiler	double precision	cost/processor	\$/TSpS
700 MHz Athlon	gcc-3.2	37s	\$625	3030
733 MHz Pentium III	egcs-2.91.66	123s	\$2,930	18170
1.4 GHZ Athlon MP	gcc-2.96	22s	\$750	2240

5.2.2 OpenMP performance tuning

In figure 5.3, the same test case with the improved fine-grain parallelization is shown. The removal of the barriers gives a much better speedup and provides a better performance of the fine-grain parallelization. The program scales better for larger number of shared processors. However, there still seems to have a limit on the number of the processor used for the fine-grain parallelization. Since the memory is distributed in planes of constant k over the available processors using more than 32 processors is no longer advantageous. By choosing a much larger test case of 256^3 grid points, as also shown in figure 5.3 by the dashed line, it can be seen that because of a larger computational load per processor the performance scales much better. For this case, the program performs well up to 64 processors.

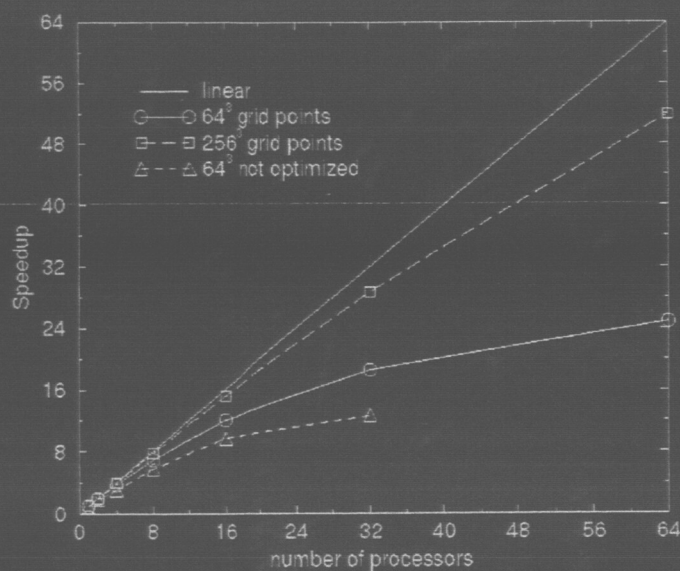


Figure 5.3: Speedup with reduced synchronization and orphaned OpenMP directives

5.2.3 Parallel performance

In order to achieve a reasonable throughput the wall time for our computations with LESTool must be at most on the order of 10s / time step. A comparison of the wall times for the three different test cases follows

Comparison of wall times

In figure 5.4 the wall times are compared for KLAT2 and KFC1. The performance of KFC1 is obviously better than that of KLAT2 because of the higher clock frequency. The clock frequency of KFC1 is double than that of KLAT2 but the speed of LESTool on KFC1 is obviously not twice compared to that on KLAT2. Here, the memory system plays an important role as well—KFC1 has a PC21000 memory system, whereas KLAT2 has a PC100 memory system. Our goal to reach a time of less than 10s / time step is already reached on two processors on KFC1, while for KLAT2 it takes about six processors to come into the same total performance range

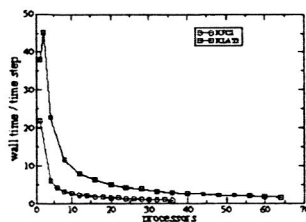
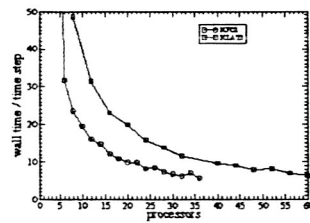
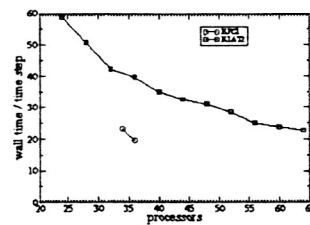


Figure 5.4: Wall clock time on KLAT2 and KFC1 for the 64^3 case

The 128^3 case is much more demanding. In this case it takes about ten nodes or twenty processors on our dual processor machine KFC1 to obtain a wall clock time under 10s. For KLAT2 forty processors are needed to obtain the desired computational speed. Note that for this example KFC1 does achieve twice the speed of KLAT2.

The 196^3 case severely stresses our cluster computers. Here we cannot reach the 10s mark on either of the clusters. For KFC1, we can achieve less than 20s / time step on 36 processors, or twice the desired speed. With KLAT2 the best wall time achievable is 22.7s.

Figure 5.5: Wall clock time on KLAT2 and KCF1 for the 128^3 caseFigure 5.6: Wall clock time on KLAT2 and KCF1 for the 196^3 case

Speedup on different network architectures

KFC1 has a three-way channel-bonded network as described in section ?? . The code scales well for all three test cases as shown in figure 5.7. Note that we could run the single processor version for the 128^3 but the performance was so low because of swapping that we decided to scale the larger cases to the minimum number of processors on which the case could reasonably run. Note that the sizeable drop in performance for 34 processors in all three cases. This is related to poor load balance. The only way we can divide our grid is in two slices in j-direction and seventeen slices in the k-direction. Given the cubical symmetry of this problem, the best subdivision is an equal number of cuts in the i, j, k direction. An example for an optimal partitioning is the case for 16 processors. Each subblock consists of 16^3 grid points. This example shows the importance of the partitioning on the parallel performance of LESTool.

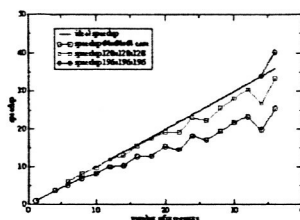


Figure 5.7: Speedup on KFC1 for the 64^3 , 128^3 and the 196^3 cases

KLAT2 has a FNN network architecture which is optimized for next neighbor communication as described in section ?? . LESTool scales even better on KLAT2 than it does on KFC1 for all three test cases as shown in figure 5.8. Note that a similar effect caused by uneven load splitting can be seen on this machine as well.

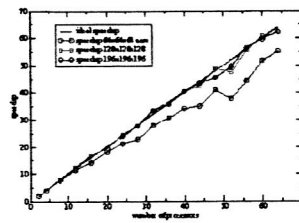


Figure 5.8: Speedup on KLAT2 for the 64^3 , 128^3 and the 196^3 cases

Chapter 6

DNS of homogeneous turbulence

6.1 Creation of initial conditions

Chapter 7

DNS of turbulent channel flow

The results presented in this paper were based on a direct numerical simulation of a fully developed channel flow reported in [?]. The flow has two periodic (x and z) directions and the solid wall condition was imposed in the y direction. The Reynolds number based on the wall shear velocity, $u_\tau = \sqrt{\tau_w/\rho}$, is $Re_\tau = \rho u_\tau H/\mu_w = 180$, where H is half the channel width. The streamwise and spanwise dimensions of the channel are $4\pi H$ and $2\pi H$, respectively. The computation is carried out on a grid using $200 \times 121 \times 200$ grid points in x , y , and z , respectively. The flow field is initialized with a laminar solution and random fluctuations are superimposed on the pressure field. The governing equations are then integrated in time until a statistical equilibrium is reached ($t u_\tau/H > 30$).

Figure 7.1 shows the dimensionless velocity, $u^+ = u/u_\tau$, plotted against dimensionless wall distance, $y^+ = y u_\tau/\nu$. Also shown in this figure is the comparison of the predicted mean velocity profile against the data cited in [?] for the same Reynolds number. The dotted line represents the desirable profile in the viscous sublayer, $u^+ = y^+$, and the dashed line denotes the log-law line, $u^+ = \ln(y^+)/0.41 + 5.2$. The figure shows that the current mean velocity profile matches the data of Kim et al. [?] very well. The comparison of the rms values as shown in figure 7.2 of the velocity shows good agreement with the data of Kim et al.

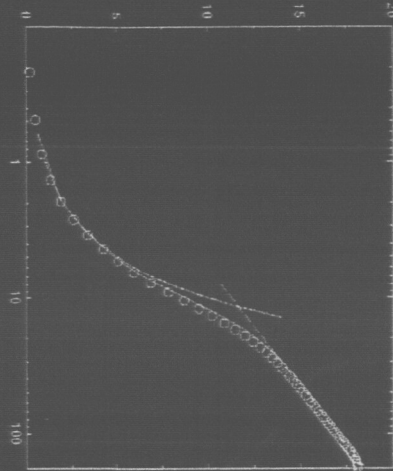


Figure 7.1: Mean velocity profiles: —, LESTool; $\circ$ $\circ$ $\circ$, DNS of [?], $\cdots$, viscous sublayer, $---$, log layer.

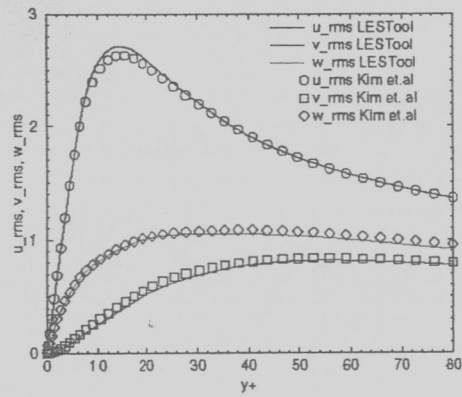


Figure 7.2: Comparison of RMS values

Chapter 8

Cylinder Flow

Chapter 9

Cascade simulations

For the cascade a series of grids are used. The finest resolution is $400 \times 200 \times 200$. A 2d view of the grid is given in figure 9.1.

Figure 9.1: 2D view of the grid

A 3D view of the grid is given in figure 9.2.

In the inflow plan (grey plane in figure 9.2) a random pressure field is prescribed to simulate the unsteady turbulent inflow. The blade, shown in red has wall boundary conditions and the green plane shows the outflow condition. The simulation is initialized with a homogeneous turbulence field as shown in figure



Figure 9.2: 3D view of the grid

9.3.

9.1 $Re = 25000$

9.2 $Re = 50000$

In figure 9.4 the averaged solution in a plane is shown.



Figure 9.3: Contour surface of initial velocity disturbance

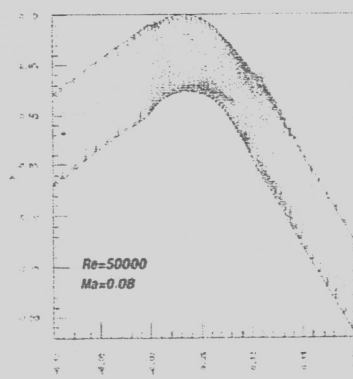
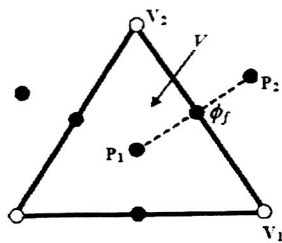
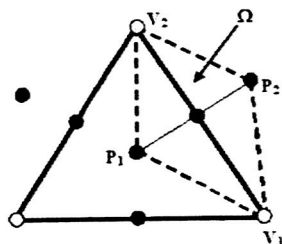


Figure 9.4: Contour surface of initial velocity disturbance

¹⁸Inoue, O. and Yamazaki, T., "Secondary vortex streets in two-dimensional cylinder wakes," Fluid Dynamics Research, vol. 25, 1999, pp. 1-18.



(a)



(b)

Fig. 1 Schematic diagrams for integration areas. (a) convective fluxes, and (b) diffusive fluxes.

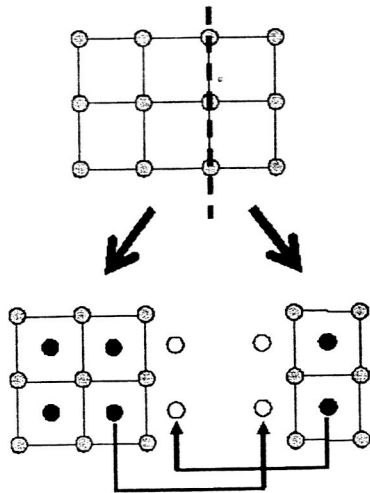


Fig. 2 A schematic diagram of cell-centered partitioning approach

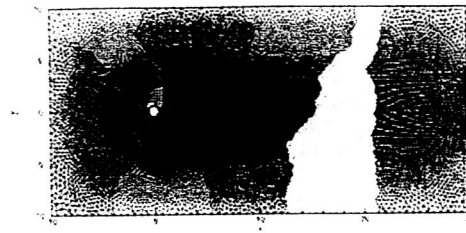


Fig. 3(a) Partitioned triangular mesh for 2D flow over a circular cylinder

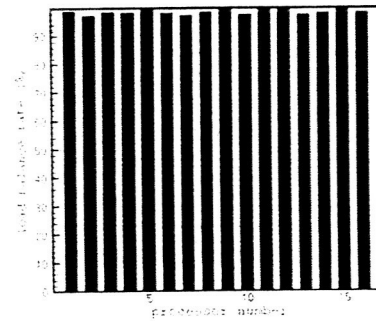


Fig. 3(b) Load-balance distribution on each node in parallel computation

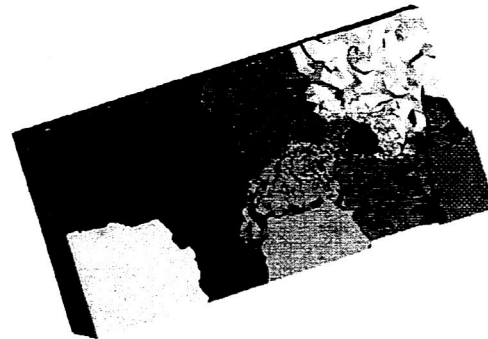


Fig. 3(c) Partitioned tetrahedral mesh for 3D flow over a circular cylinder

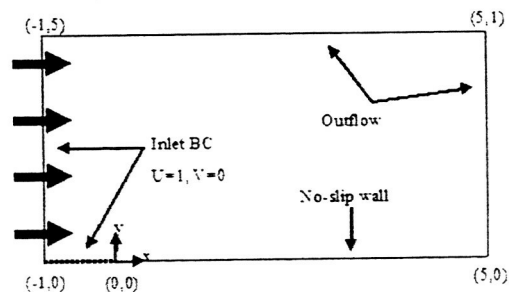


Fig. 4 Schematic diagram of laminar incompressible flow past flat plate

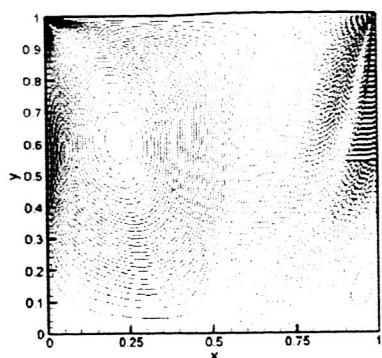


Fig. 8(b) The v -velocity contour plot for two-dimensional driven cavity flow at $Re=400$.

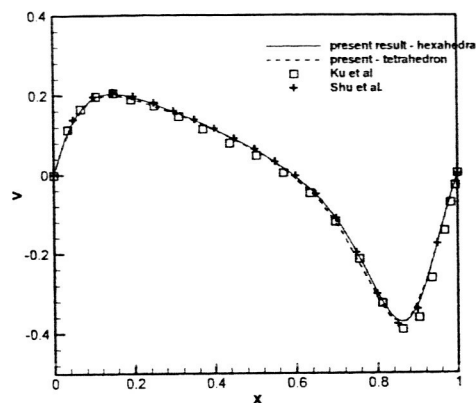


Figure 9(b) The v -velocity profile at the vertical centerline of $z=0.5$ plane with present results, Ku's and Shu's results.

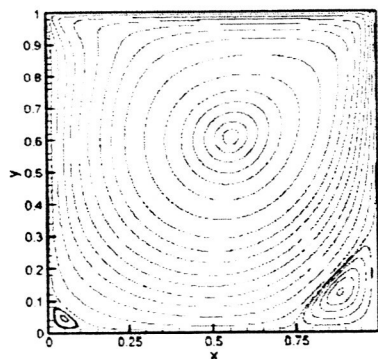


Fig. 8(c) Streamline plot for two-dimensional driven cavity flow at $Re=400$.

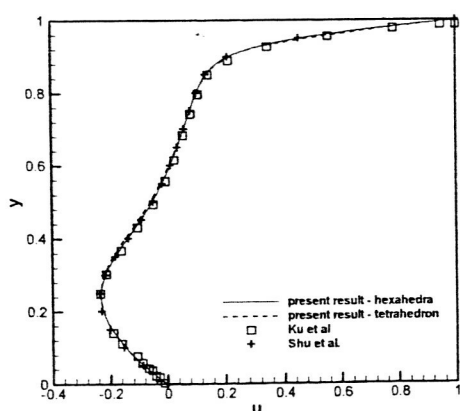
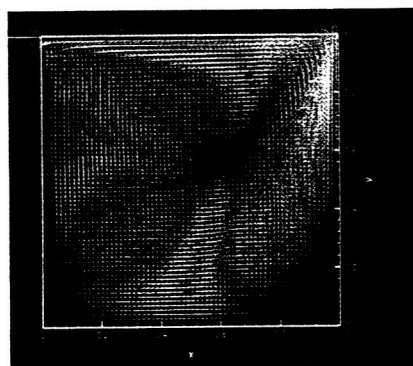
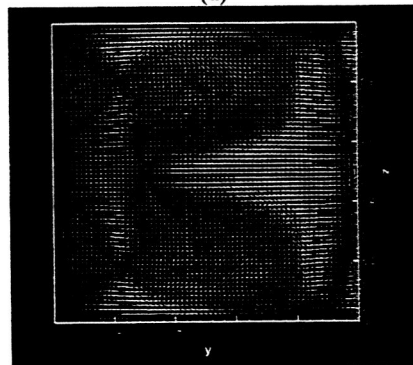


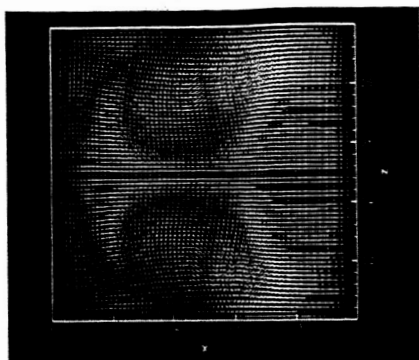
Figure 9(a) The u -velocity profile at the horizontal centerline of $z=0.5$ plane with present results in hexahedral and tetrahedral meshes, and also Ku's and Shu's results.



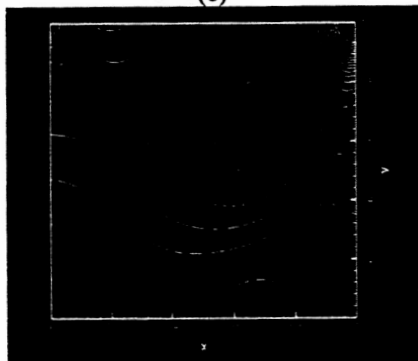
(a)



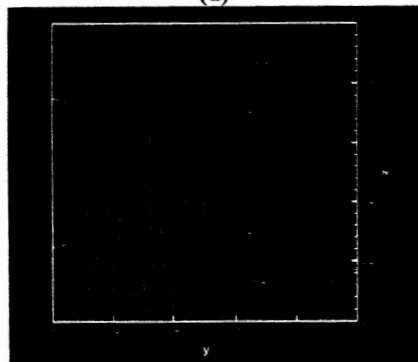
(b)



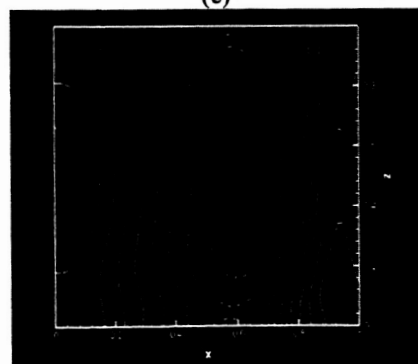
(c)



(d)



(e)



(f)

Figure 10(a)-(c) The velocity vector and (d)-(f) pressure contours from present results with hexahedral mesh at $z=0.5$, $x=0.5$, and $y=0.5$ planes respectively.

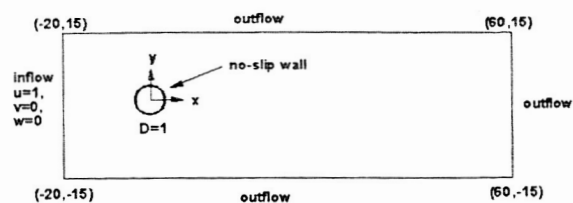
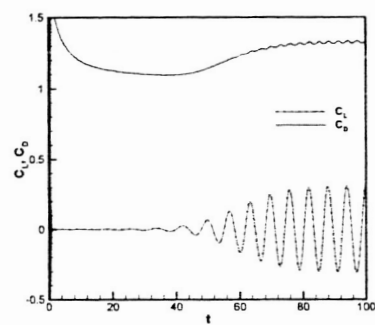
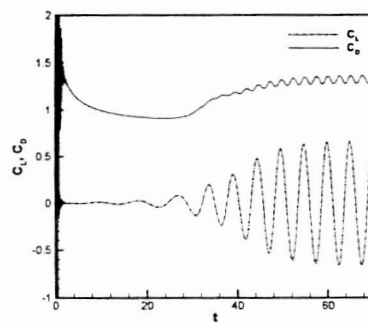


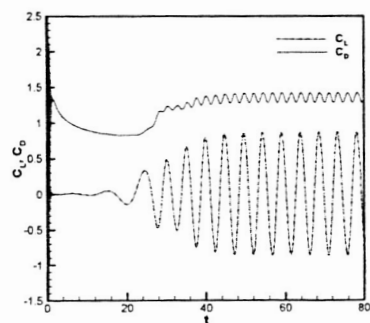
Fig. 11 Schematic diagram of flow over a circular cylinder with dimensions and boundary conditions



(a)



(b)



(c)

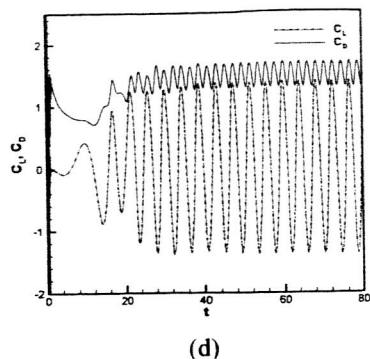
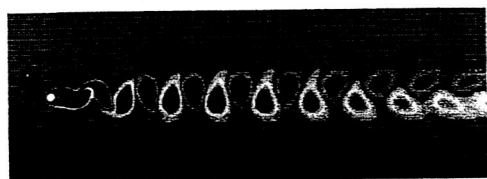
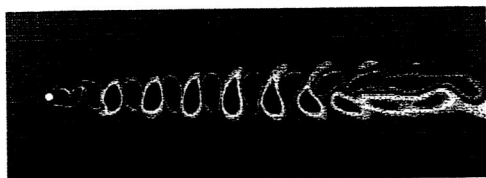


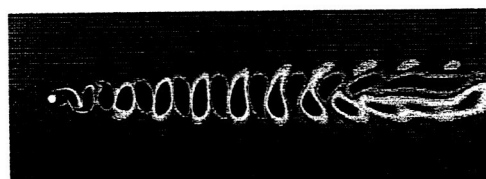
Fig. 12 C_L and C_D history plots of two-dimensional simulation for (a) $Re=100$, (b) $Re=200$, (c) $Re=300$, and (d) $Re=1000$.



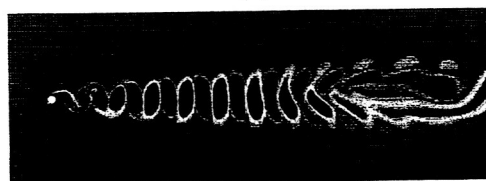
(a)



(b)

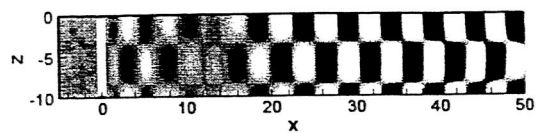


(c)

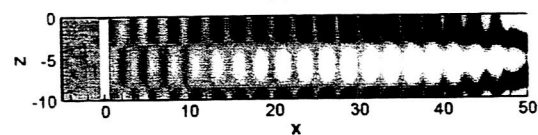


(d)

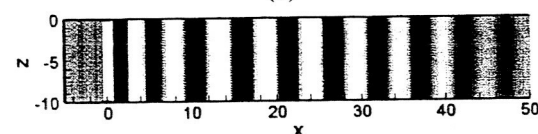
Fig. 13 Vorticity contours for two-dimensional simulation. (a) $Re=100$, (b) $Re=200$, (c) $Re=300$, and (d) $Re=1000$.



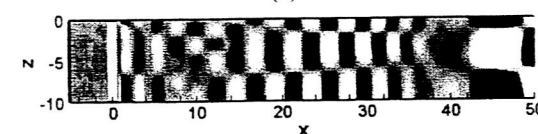
(a)



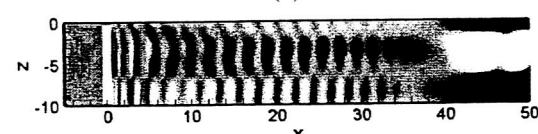
(b)



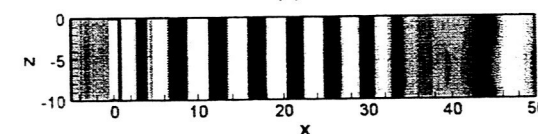
(c)



(d)



(e)



(f)

Fig. 14 Vorticity component contours for three-dimensional simulation at $y=0$ plane, where (a)-(c) ω_x , ω_y , and ω_z for $Re=100$ and (d)-(f) ω_x , ω_y , and ω_z for $Re=200$.

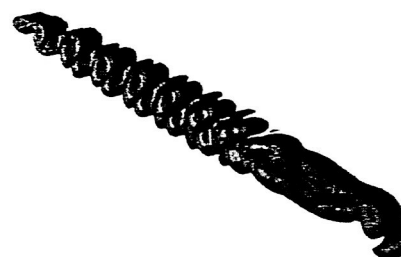


Fig. 15 The iso-surface of vorticity magnitude for $Re = 200$ from present three-dimensional simulation result.

Appendix II - UNCLE

A Center Pressure Based Method for Two/Three-Dimensional Unstructured Incompressible Navier-Stokes Solver

H. Chen, P. G. Huang, and R. P. LeBeau

University of Kentucky

Dept. of Mechanical Engineering

Lexington, KY 40506

Abstract

A center pressure based method is presented in this paper, and which has been implemented into a new two/three-dimensional parallel unstructured CFD code, UNCLE, which is developed at the University of Kentucky to meet the challenges of physical problems with complex geometries and complicated boundary conditions while maintaining high computational efficiency. Good load balancing across computational nodes is achieved by using METIS. In order to demonstrate the accuracy and performance of center pressure based method, several test cases are presented for validation such as two-dimensional incompressible flow past a flat plate, two/three-dimensional driven cavity flow, and two/three-dimensional flow over a circular cylinder. Notably, an extensive qualitative and quantitative study of two-dimensional flow over a circular cylinder for low Reynolds number is also presented in this paper.

1 Introduction

Continual improvements in computer technologies and computational fluid dynamics (CFD) algorithms have established CFD codes as a reliable tool for fundamental research or industrial applications. To deal with increasing grid sizes and demands for faster output, parallel computation of CFD has become a standard approach. To deal with the different challenge presented by some physical problems with complex geometries and complicated boundary conditions is now approached through unstructured CFD grids due to their ability to smoothly conform to complicated boundaries. However, combining unstructured grids with a parallel code presents still other challenges, such as achieving well-balanced grid decomposition on a distributed system and efficient parallel performance. In order to meet these challenges, a center pressure based method has been implemented into a new parallel unstructured CFD code called UNCLE, which has been developed at the University of

Kentucky. UNCLE is designed to meet the challenges of using unstructured grid codes on high-performance parallel computers. It is a two/three-dimensional finite volume unsteady incompressible Navier-Stokes solver with center pressure based SIMPLE algorithm with second order accuracy in both time and space. To increase flexibility in complex geometries, center pressure based method is extended to use a variety of grid types, such as triangular, quadrilateral, tetrahedral, and hexahedral meshes. To obtain good load balancing across computational nodes, METIS [1] is applied for domain decomposition. METIS is a set of programs for partitioning graphs and finite element meshes, and for producing fill-reducing orderings for sparse matrices. The algorithms implemented in METIS are based on multilevel graph partitioning schemes. The key features of METIS include extremely fast partition, high quality partitions, and low fill orderings. The parallel construction of UNCLE is based on message passing interface (MPI) protocols and has worked successfully on systems ranging from commodity PC clusters up to traditional supercomputers. In order to demonstrate the accuracy and performance of center pressure based method, several test cases are presented for validation such as two-dimensional incompressible flow past a flat plate, two/three-dimensional driven cavity flow, and two/three-dimensional flow over a circular cylinder.

2 Numerical Methods

A center pressure based method for two/three-dimensional finite volume unstructured incompressible Navier-Stokes solver for steady/unsteady flow fields is presented in this paper. It is center pressure based SIMPLE algorithm with second order accuracy in both time and space. In order to compute numerical flux on interfaces, a second order upwind scheme is adopted to compute advection terms and second order central difference scheme is used for diffusion terms. Non-staggered grids with the Rhie and Chow momentum interpolation method [2] is employed to correct the checkerboard solution in the SIMPLE scheme.

2.1 Governing equations

The governing equations for unsteady incompressible viscous flow under the assumption of no body force and heat transfer are as below.

Conservation of Mass

$$\frac{\partial}{\partial t} \int_V \rho dV = - \oint_S \rho u_i n_i dS \quad (1)$$

Conservation of Momentum

$$\frac{\partial}{\partial t} \int_V \rho u_j dV = - \oint_S \rho u_i n_i u_j dS - \oint_S p n_j dS + \oint_S \tau_{ij} n_i dS \quad (2)$$

Conservation of Energy

$$\frac{\partial}{\partial t} \int_V \rho E dV = - \oint_S \rho u_i n_i E dS - \oint_S p u_j n_j dS + \oint_S u_j \tau_{ij} n_i dS \quad (3)$$

where ρ is density, p is pressure, u_i is component of velocity vector, n_i is unit normal vector of the interface, τ_{ij} is tensor of shear force, and specific internal energy $E = e + \frac{1}{2}(u^2 + v^2 + w^2)$. Notably, density is constant for incompressible flow.

2.2 Convective and diffusive fluxes

Figure 1(a) shows the schematic diagram for integration area for convective fluxes. By using Taylor series expansion, flow properties on the interface can be obtained by Eq. (3).

$$\begin{aligned} \phi^{LHS} &= \phi_{P_1} + \frac{\partial \phi}{\partial x} \Big|_{P_1} (x_f - x_{P_1}) \\ &\quad + \frac{\partial \phi}{\partial y} \Big|_{P_1} (y_f - y_{P_1}) + \frac{\partial \phi}{\partial z} \Big|_{P_1} (z_f - z_{P_1}) + HOT \\ \phi^{RHS} &= \phi_{P_2} + \frac{\partial \phi}{\partial x} \Big|_{P_2} (x_f - x_{P_2}) \\ &\quad + \frac{\partial \phi}{\partial y} \Big|_{P_2} (y_f - y_{P_2}) + \frac{\partial \phi}{\partial z} \Big|_{P_2} (z_f - z_{P_2}) + HOT \end{aligned} \quad (3)$$

where ϕ stands for the velocity components and pressure, the superscript *RHS* and *LHS* denote the approximation from the right-hand side and left-hand side of the interface respectively, and *HOT* represents higher order terms. By substituting Eq. (3) into Eq. (4), interfacial flow properties ϕ_f can be obtained.

$$\phi_f = \frac{1}{2}(\phi^{RHS} + \phi^{LHS}) - \frac{1}{2} \text{sign}(1, \hat{n})(\phi^{RHS} - \phi^{LHS}) \quad (4)$$

The gradients at the nodal points (cell centers) are evaluated by the Gauss's divergence theorem as below.

$$\begin{aligned} \int_V \frac{\partial \phi}{\partial x_i} dV &= \int_A \phi n_i dA \\ \frac{\partial \phi}{\partial x_i} &\approx \frac{\sum_{k=1}^{N_{face}} \phi n_{i,k} A_k}{V} \end{aligned} \quad (5)$$

where N_{face} is the total number of interfaces of the cell and V denotes the volume of the control volume cell.

The schematic diagram for diffusive fluxes is shown in Fig 1(b). The gradients at the interface can be evaluated by using Chain rule as Eq. (6).

$$\begin{aligned} \frac{\partial \phi}{\partial x} &= \frac{\partial \phi}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial \phi}{\partial \eta} \frac{\partial \eta}{\partial x} + \frac{\partial \phi}{\partial \zeta} \frac{\partial \zeta}{\partial x} \\ \frac{\partial \phi}{\partial y} &= \frac{\partial \phi}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial \phi}{\partial \eta} \frac{\partial \eta}{\partial y} + \frac{\partial \phi}{\partial \zeta} \frac{\partial \zeta}{\partial y} \\ \frac{\partial \phi}{\partial z} &= \frac{\partial \phi}{\partial \xi} \frac{\partial \xi}{\partial z} + \frac{\partial \phi}{\partial \eta} \frac{\partial \eta}{\partial z} + \frac{\partial \phi}{\partial \zeta} \frac{\partial \zeta}{\partial z} \end{aligned} \quad (6)$$

where the local coordinate system (ξ, η, ζ) is defined by the type of mesh separately.

For triangular mesh in Fig. 1(b), ξ is the vector from nodal point P_1 to P_2 , η is the vector from vertex V_1 to V_2 , and Ω is the integration area for diffusive fluxes. The diffusive fluxes can be approximated by Eq. (7).

$$\begin{aligned} \left(\frac{\partial \phi}{\partial x} \right)_f &\approx \frac{1}{2\Omega} [(\phi_{P_2} - \phi_{P_1})(y_{P_2} - y_{P_1}) - (\phi_{V_2} - \phi_{V_1})(y_{V_2} - y_{V_1})] \\ \left(\frac{\partial \phi}{\partial y} \right)_f &\approx \frac{-1}{2\Omega} [(\phi_{P_2} - \phi_{P_1})(x_{P_2} - x_{P_1}) - (\phi_{V_2} - \phi_{V_1})(x_{V_2} - x_{V_1})] \end{aligned} \quad (7)$$

where ϕ_{P_i} denotes the properties at nodal points and ϕ_{V_i} denotes the properties at vertices.

The values of vortices are obtained by averaging surrounding nodal value, in which inverse distances from all surrounding nodal points are considered as weighted function.

2.3 Center pressure based SIMPLE algorithm

By using an initial pressure field, P^n , we can obtain u^n , v^n , and w^n by solving the momentum equations in an uncoupled form. The momentum equations can be written in the form as Eq. (8).

$$\begin{aligned} a_c \Delta u &= \sum_{nb} a_{nb} \Delta u + RHS_u \\ a_c \Delta v &= \sum_{nb} a_{nb} \Delta v + RHS_v \\ a_c \Delta w &= \sum_{nb} a_{nb} \Delta w + RHS_w \end{aligned} \quad (8)$$

where the coefficients a_{nb} and a_c are

$$a_{nb} = \max(-\dot{m}_f, 0) + \mu_f (\xi_x n_1 + \xi_y n_2 + \xi_z n_3) A,$$

$$a_c = \sum_{nb} a_{nb},$$

and the *RHS* term can be written as

$$\begin{aligned} RHS_u &= - \sum_{i=1}^{N_{face}} [\dot{m}_i u_i^n - (\tau_{11}^n - p^n)_i n_{i,1} A_i - (\tau_{21}^n)_i n_{i,2} A_i - (\tau_{31}^n)_i n_{i,3} A_i] \\ &= - \sum_{i=1}^{N_{face}} [\dot{m}_i u_i^n - (\tau_{11}^n)_i n_{i,1} A_i - (\tau_{21}^n)_i n_{i,2} A_i - (\tau_{31}^n)_i n_{i,3} A_i] - \frac{\partial p^n}{\partial x} V_c \\ RHS_v &= - \sum_{i=1}^{N_{face}} [\dot{m}_i v_i^n - (\tau_{12}^n)_i n_{i,1} A_i - (\tau_{22}^n - p^n)_i n_{i,2} A_i - (\tau_{32}^n)_i n_{i,3} A_i] \\ &= - \sum_{i=1}^{N_{face}} [\dot{m}_i v_i^n - (\tau_{12}^n)_i n_{i,1} A_i - (\tau_{22}^n)_i n_{i,2} A_i - (\tau_{32}^n)_i n_{i,3} A_i] - \frac{\partial p^n}{\partial y} V_c \\ RHS_w &= - \sum_{i=1}^{N_{face}} [\dot{m}_i w_i^n - (\tau_{13}^n)_i n_{i,1} A_i - (\tau_{23}^n)_i n_{i,2} A_i - (\tau_{33}^n - p^n)_i n_{i,3} A_i] \\ &= - \sum_{i=1}^{N_{face}} [\dot{m}_i w_i^n - (\tau_{13}^n)_i n_{i,1} A_i - (\tau_{23}^n)_i n_{i,2} A_i - (\tau_{33}^n)_i n_{i,3} A_i] - \frac{\partial p^n}{\partial w} V_c \end{aligned}$$

where subscript *c* denotes the cell we are solving, subscript *nb* denotes the neighbor cells, and *A* denotes the interfacial area. In this paper, we solve Eq. (8) by using Gauss-Seidel method. Then, we can obtain u^* , v^* , and w^* by Eq. (9).

$$\begin{aligned} u^* &= u^n + \Delta u \\ v^* &= v^n + \Delta v \end{aligned} \quad (9)$$

$$w^* = w^n + \Delta w$$

Although at this stage u^* , v^* , and w^* satisfy the momentum equations, they do not necessarily satisfy the continuity equation. In order to satisfy the mass conservation, one has to interpolate the velocity to the interface.

However, this interpolation will lead to the checkerboard solutions. In order to avoid the checkerboard solutions, one has to allow the interfacial velocity to be driven solely by the pressure difference. To achieve this aim without sacrificing the accuracy, one can divide the interpolated interfacial velocity into two components: one is the velocity component without the pressure contribution and the other is solely the pressure contribution. The former, which is at the cell center, can be written as:

$$\begin{aligned} \tilde{u}^* &= u^* + \frac{\partial p^n}{\partial x} \frac{V_c}{a_c} \\ \tilde{v}^* &= v^* + \frac{\partial p^n}{\partial y} \frac{V_c}{a_c} \\ \tilde{w}^* &= w^* + \frac{\partial p^n}{\partial z} \frac{V_c}{a_c} \end{aligned} \quad (10)$$

The latter is obtained directly from the pressure difference of the two adjacent nodal points, P_1 and P_2 such that the interfacial velocity can be expressed as:

$$\begin{aligned} u_f^* &= \tilde{u}_f^* - \left(\frac{\partial p^n}{\partial x} \right)_f \frac{V_f}{a_f} \\ v_f^* &= \tilde{v}_f^* - \left(\frac{\partial p^n}{\partial y} \right)_f \frac{V_f}{a_f} \end{aligned} \quad (11)$$

$$w_f^* = \tilde{w}_f^* - \left(\frac{\partial p^n}{\partial z} \right)_f \frac{V_f}{a_f}$$

Where V_f and a_f are obtained by interpolation to the interface.

We further assume that there are corrections to u_f^* , v_f^* , and w_f^* , such that the continuity equation can be satisfied by using Eq. (12).

$$\sum_{i=1}^{N_{face}} \rho [(u_f^* + \Delta u_f') n_1 + (v_f^* + \Delta v_f') n_2 + (w_f^* + \Delta w_f') n_3] A = 0 \quad (12)$$

We can rewrite Eq. (12) as

$$\sum_{i=1}^{N_{face}} \rho [\Delta u_f' n_1 + \Delta v_f' n_2 + \Delta w_f' n_3] A = - \sum_{i=1}^{N_{face}} \rho [u_f^* n_1 + v_f^* n_2 + w_f^* n_3] A \quad (13)$$

where the right-hand side in Eq. (13) represents the mass imbalance in the control volume cell.

One assumes there is a corresponding pressure correction field, p' , which drives the velocity corrections according to:

$$\begin{aligned} \Delta u_f' &\approx - \left(\frac{\partial p'}{\partial x} \right)_f \frac{V_f}{a_f} \approx - \frac{V_f}{a_f} \xi_x (p'_{P_2} - p'_{P_1}) \\ \Delta v_f' &\approx - \left(\frac{\partial p'}{\partial y} \right)_f \frac{V_f}{a_f} \approx - \frac{V_f}{a_f} \xi_y (p'_{P_2} - p'_{P_1}) \\ \Delta w_f' &\approx - \left(\frac{\partial p'}{\partial z} \right)_f \frac{V_f}{a_f} \approx - \frac{V_f}{a_f} \xi_z (p'_{P_2} - p'_{P_1}) \end{aligned} \quad (14)$$

By substituting the velocity correction equations into the equation for the mass imbalance, we can obtain the equations of the pressure correction:

$$a_c p'_c = \sum_{nb} a_{nb} p'_{nb} + b \quad (15)$$

where a_{nb} and a_c in the continuity equation are

$$a_{nb} = \rho \left[\frac{V_f}{a_f} \xi_x n_1 + \frac{V_f}{a_f} \xi_y n_2 + \frac{V_f}{a_f} \xi_z n_3 \right] A,$$

$$a_c = \sum_{nb} a_{nb},$$

and

$$b = - \sum_{nb} \rho [u_f^* n_1 + v_f^* n_2 + w_f^* n_3] A.$$

Once the pressure correction is obtained, one can update the pressure field by:

$$p^{n+1} = p^n + \alpha_p p' \quad (16)$$

where α_p is the under-relaxation factor for pressure and is generally with a value of 0.5-0.8. Then the velocity

correction on the interfaces as well as nodal points will be updated.

2.4 Partitioning approach

Figure 2 shows the schematic diagram of partitioning approach for center pressure based method, which is also implemented into UNCLE. In Fig. 2, blue points indicate vertices, red points indicate nodal points, and white points indicate the boundary points. By using this approach, the control volumes on the boundary are not split. Only communication of nodal values is needed for parallel computation which makes the implement of MPI in an unstructured grid more straightforward.

Excellent load balancing between the subgrids on each node is achieved through using METIS for domain decomposition. METIS can partition an unstructured grid into any integer number of zones without losing load balance. It is compatible with many platforms, convenient for running CFD codes on a variety of supercomputer to cluster architectures. Present partitioning approach has been tested by a number of two/three-dimensional geometries. All results show good load balances. In order to demonstrate the capability of this partitioning approach, this paper will present two cases—one is the grid for two-dimensional flow over circular cylinder in triangular mesh, and the other is the grid for three-dimensional flow over a circular cylinder in tetrahedron mesh. The definition of load-imbalance rate L_{IMB} and load-balance rate L_B in this paper is defined as Eq. (17) and (18).

$$L_{IMB} = \frac{|N_{node} - N_{avg}|}{N_{avg}} \times 100\% \quad (17)$$

$$L_B = 1 - L_{IMB} \quad (18)$$

where N_{node} is grid size of the node and N_{avg} is the average grid size. By using load balance rate, we can compare the load balance quantitatively.

Figure 3(a) shows the partitioned grid for 2D flow over a circular cylinder in triangular mesh. The number of total grid points is 51,363 and the number of total cells is approximately 0.1M. The grid is partitioned to 16 zones for parallel computation. The cell distribution is not uniform, denser near the cylinder and coarser away. The load-balance distribution on each node is shown in Fig. 3(b). The x-axis indicates the node number and the y-axis indicates load-balance rate. The resulting load-balance rates are very close to 100% on every node with an average load balance rate of 98.37%. Figure 3(c) shows a partitioned grid for three-dimensional flow over circular cylinder with the internal grid distribution visible in a cut-away. The number of total grid points is approximately 0.3M and the number of total cells is approximately 1.3M. In this case, the grid is partitioned to 32 zones. The average load-balance rate is 97.9%. Our test results show that present partitioning approach

has excellent load balance in two/three-dimensional grids with various types of meshes by using METIS for domain decomposition.

2.5 Time discretization

In this paper, a second-order fully implicit scheme is employed for the temporal discretization. In here, we take a one-dimensional equation example:

$$\frac{3\phi^{n+1} - 4\phi^n + \phi^{n-1}}{2\Delta t} + \frac{\partial f(\phi^{n+1})}{\partial x} = 0 \quad (19)$$

where ϕ is primitive variable, f is interfacial flux, and the superscript n indicates the index in time. A deferred iterative algorithm is employed to obtain ϕ^{n+1} by substituting (20) into (19),

$$(\phi^{n+1})^m = (\phi^{n+1})^{m-1} + (\Delta\phi)^m \quad (20)$$

where the subscript m stands for the sub iteration level. The final equation is

$$\frac{3(\Delta\phi)^m}{2\Delta t} + \frac{\partial f(\Delta\phi)^m}{\partial x} = \frac{(\phi^n - \phi^{n-1})}{2\Delta t} - \frac{3((\phi^{n+1})^{m-1} - \phi^n)}{2\Delta t} - \frac{\partial f((\phi^{n+1})^{m-1})}{\partial x} \quad (21)$$

The right-hand-side of Eq. (21) is explicit and can be implemented in a straightforward manner to discretize the spatial derivative term. The deferred iterative algorithm is strongly stable, and the solution ϕ^{n+1} is obtained by using inner iterations to reach the convergent solution of the right-hand-side of Eq. (21), which means $\Delta\phi$ is approximate to zero. A sub-iteration is performed at every time step so that this method is fully implicit.

3 Results

3.1 Two-dimensional laminar incompressible flow past a flat plate

In this case, two-dimensional laminar incompressible flow past a flat plate is simulated. The schematic diagram including geometry and boundary conditions is shown in Fig. 4. A uniform free stream boundary condition is imposed on the inlet. Because of the viscous effect, the laminar boundary layer begins to grow at the position $x=0$ on the plate. The initial condition for the entire computational domain is uniform free stream. A quadrilateral mesh of 600x200 is used for both Reynolds number (Re) at 5,000 and 50,000 based on a non-dimensional length scale of unity. The minimum distance from the wall is 5×10^{-5} (corresponding to $y^+ = 0.1$) which is fine enough to capture the phenomena within the boundary layer. In this case, the computational domain is divided into 16 zones. The parallel computation is performed on KFC3a, a 16 nodes PC cluster with Pentium IV 2.4

GHz CPU and gigabit network developed at the University of Kentucky. The results of this computation are compared to the standard Blasius solution for a laminar flat-plate boundary layer. Figure 5(a) shows the $\eta = y\sqrt{\text{Re}_x}/x$ versus u/U plot at $x=4.5$ from present results for $\text{Re}=5,000$ and $50,000$ in comparison with Blasius solution. Both present results are in good agreement with Blasius solution. Figure 5(b) shows the plot of momentum thickness (Re_θ) versus coefficient of skin friction (c_f) with present results and Blasius solution. Good agreement is obtained by comparing our present results with Blasius solution.

3.2 Two-dimensional driven cavity flow

In this section, two-dimensional incompressible flow in a square cavity at a Reynolds number of 400 is simulated. The fluid in the cavity is driven by a moving top with constant speed. Because driven cavity flow lacks an exact solution, an existing accurate numerical solution for this problem is used as a benchmark for comparing our results. Ghia et al. [3] presented numerical studies using the vorticity-stream function formulation for solutions up to $\text{Re}=10,000$ with 257×257 grid points, and these simulation results have been widely used as a benchmark for the driven cavity problem. The schematic diagram of this case with geometry and boundary conditions is shown in Fig. 6. The initial condition for the entire computational domain is stationary everywhere. In order to compare Ghia's results, the number of grid points used is 257×257 or $66,049$ and $65,536$ cells are used in a quadrilateral mesh. For a triangular mesh, $66,546$ cells, which is approximately the same as quadrilateral mesh, and $33,618$ grid points are used in our computation. Figure 7(a) shows the u -velocity profile along the horizontal center line for both present results with quadrilateral and triangular mesh and Ghia's result. Both present results are in good agreement with Ghia's result. It also shows the present solution is identical and is independent of mesh types. Figure 7(b) shows the u -velocity profile along the horizontal center line; again, the results match. Figure 8 presents the u - and v -velocity contours and streamline plot from the present results on the quadrilateral mesh. In Fig. 8(a), the u -velocity contours range from -0.33 to 1.0 with 100 intervals. Solid lines indicate the positive values and dashed lines negative ones. In Fig. 8(b), the v -velocity contours range from 0.64 to 0.31 with 100 intervals. The flow structures including the location of the major vortex center, the bubble in the right bottom corner, and a small bubble in the left bottom corner are shown clearly in Figure 8(c), and are in good agreement with the results of Ghia.

3.3 Three-dimensional driven cavity flow

The three-dimensional version of the preceding problem is also a standard case for a new flow solver. In 1987, Ku et al. [4] simulated three-dimensional flow in a cubic cavity by using pseudospectral methods to solve the Navier-Stokes equations for $\text{Re} = 100, 400$, and 1000 . In 2003, Shu et al. [5] repeated this problem by using the SIMPLE algorithm with the differential quadrature (DQ) method. They simulated the three-dimensional driven cavity flow at $\text{Re} = 100, 200, 400$ and 1000 and compared their results with Ku's results. In this section, we simulated this problem at $\text{Re} = 400$ and compared our results with those of Ku and Shu. In order to validate the results with different meshes, two grids are used to study this problem. One is a hexahedral mesh with $67 \times 67 \times 67$ grid points and $287,496$ cells, and the other is a tetrahedral mesh with $79,951$ grid points and $446,953$ cells. The geometry of this problem is a unit cube. The boundary condition at the $y = 1$ plane is uniform flow with $u=1, v=0$, and $w=0$, and all other boundary conditions are no-slip walls. The initial condition for the entire computational domain is stationary. Figure 9(a) shows the u -velocity profile at the horizontal centerline of the $z = 0.5$ plane for the present hexahedral and tetrahedral mesh results as well as those of Ku and Shu. Figure 9(b) shows the v -velocity profile at the vertical centerline of $z = 0.5$ for the same set of simulations. Both present simulations show essentially identical solutions, and both are in good agreement with Ku and Shu. Figure 10 shows the velocity vectors and pressure contours taken from the present simulation using the hexahedral mesh on the $z=0.5, x=0.5$, and $y=0.5$ planes respectively. As before, for the pressure contours dashed lines mean negative values. The established flow structures including the locations of the vortex center and the positive and negative regions in each plane are clearly visible and in good agreement with Ku's and Shu's results.

3.4 Two/Three-dimensional incompressible flow over a circular cylinder

Flow over a circular cylinder is a standard unsteady test problem. Many two-dimensional numerical studies have been done on this flow. In 1990, Rogers and Kwak [6] studied flow over a circular cylinder for $\text{Re} = 200$ using an upwind differencing scheme to solve the incompressible Navier-Stokes equations. They compared their results, which were obtained using 3rd order and 5th order upwind differencing schemes, with computational and experimental results. In 1995, Ku [7] studied this problem for Reynolds numbers $100, 250, 500$, and $1,000$ using a pseudospectral element method. Alonso et al. [8] studied the vortex shedding over a circular cylinder at $\text{Re} = 500$ with a multigrid unsteady Navier-Stokes solver with a flux-limited dissipation

scheme in 1995. In 1998, Liu et al. [9] used preconditioned multigrid methods to study unsteady incompressible flows. They investigated flow over a circular cylinder at $Re = 200$ and their results was in good agreement with other computational and experimental results. In 1999, Ronald et al. [10] reported their results for this problem at $Re = 300$ using the NASA FUN2D code. A final example of two-dimensional cylinder flow simulation is the work of Qian and Vezza [11] studied flow over a circular cylinder problem at $Re = 1000$ with a vorticity-based method in 2001.

Examples of three-dimensional numerical studies of flow over a circular cylinder are the studies of Braza and Persillon [12] and Henderson [13]. There are also many experimental studies of flow over cylinder problems such as Roshko [14], Wille [15], and Williamson [16], [17]. According to Williamson [16], the phenomena of flow over a circular cylinder can be classified by Reynolds number. Williamson found the laminar vortex shedding region to occur for Reynolds numbers between 49 and 140-194. The three-dimensional wake transition region occurs for $Re \sim 190$ to 260. For the range between $Re = 260 \sim 1000$, the three-dimensional disorder of the wake begin to increase in at the fine scales. This evolves into the shear-layer transition regime for Reynolds numbers 1,000 up to 20,000. He also noted that three-dimensional effects occur for $Re > 190$. Further detailed explanations of the remaining regimes are reported in Williamson [16]. To date, most numerical studies about this problem only focused on a single Reynolds number and either two-dimensional or three-dimensional simulations exclusively. The current results encompass Reynolds numbers 100, 200, 300, 500, 1,000, and 1500 for two-dimensional simulations and 100 and 200 for three-dimensional simulations. These results are compared with the appropriate previous studies as well as current results cross-comparisons to examine the dimensional and flow regime effects.

Figure 11 shows a schematic diagram for flow over a circular cylinder with dimensions and boundary conditions. For three-dimensional simulation, the span of the cylinder is $10D$, with D representing the reference length equal to the diameter of the cylinder. The boundary conditions at $z = 0$ and $z = -10D$ are periodic, eliminating end effects. The initial condition for the entire domain is uniform flow as inflow for all simulations and the time step is 0.005 for all cases. The grid for two-dimensional simulations is a quadrilateral mesh with 22705 cells and 22925 grid points; for three-dimensional simulation, a hexahedral mesh with 1.13M cells and 1.17M grid points is used. Both grids are densely distributed near the cylinder and wake region and coarser near the outer region. All of the two-dimensional simulations are performed on the KFC3a

cluster with 16 nodes, and all three-dimensional simulations are performed on KFC2, a 48 node commodity cluster with AMD Athlon 2000+ CPU and a channel-bonded network. It is noted that each three-dimensional simulation took approximately one week with 32 nodes on KFC2. Figure 12 presents the coefficient of Lift (C_L) and the coefficient of drag (C_D) unsteady histories of the present two-dimensional simulations for $Re = 100, 200, 300$, and 1,000 respectively. The Strouhal number (S_t) for these data sets is derived from the frequency of C_L . In our simulations, higher Reynolds number cases achieve steady Strouhal numbers faster than lower Re cases, consistent with other computational results. Table 1 presents the summary of our present two- and three-dimensional results along with other computational and experimental results. As shown in Table 1, our current results show good agreement with previous data by comparing S_t , C_L , and C_D . Fig. 13 shows the two-dimensional vorticity contours for $Re = 100, 200, 300$ and 1000. In Fig. 13, the contours are range from -0.3 to 0.3. The three-dimensional ω_z contours from the current simulations are similar to those generated by the two dimensional test cases. As seen in Fig. 13, the vortex shedding frequency increases as Re increases. For $Re = 100$, the vortices decay in the downstream. Because of the limited computational domain, we do not see the vortex structures merge to large scale vortices in our simulation. For $Re = 200, 500$ and 1000, not only does the vortex strength decay, but also the vortex structures collapse in the far downstream and start to merge to large scale vortices. This same phenomenon is reported by Inoue and Yamazaki [18]. Figure 14(a)-(c) show the present three-dimensional results of ω_x , ω_y and ω_z for $Re = 100$ and (d)-(f) for $Re = 200$ in the $y = 0$ plane. The contours of ω_x and ω_y range from -0.0005 to 0.0005 for $Re = 100$ and from -0.02 to 0.02 for $Re = 200$, and the contours of ω_z range from -0.3 to 0.3 for both cases where bright regions correspond to positive values and dark regions correspond to the negative values. Obviously, the magnitude of ω_z is much larger than the magnitudes of ω_x and ω_y . For $Re = 100$, we can see that the magnitude of ω_x decreases in the upstream and the local minimum appears at the position of 5th "roll" in Fig. 14(a). After that, the magnitude of ω_x begins to grow. In Fig. 14(b), the magnitude of ω_y grows after the flow past the cylinder. The thickness of the vortex "roll" increases downstream. From Fig. 14(c), the magnitude of ω_z decays after the flow past the cylinder due to the energy dissipation. Because the three-dimensional effects are very weak at $Re = 200$, the three-dimensional outcome is very similar to two-dimensional case. For $Re = 200$, as seen from Fig. 14(d), the magnitude of ω_x decreases in the beginning and the local minimum also appears

near the 5th "roll". In the far downstream, a transition zone is observed, after which the vortex scales transit from small to large scales. In Fig. 14(e), unlike Fig. 14(b), wavy vortex structures are observed. Figure 14(f) shows that ω_z decays after the flow past the cylinder. Figure 15 shows the iso-surface of vorticity magnitude for $Re = 200$ from the present three-dimensional simulation result. The flow structures, especially the vortex streets in the wake regions, observed in our present results are in good agreement with other simulation results.

4 Conclusions

A center pressure based method is presented in this paper, and which has been implemented successfully to a new two/three-dimensional parallel unstructured incompressible Navier-Stokes solver, UNCLE, which has been developed at University of Kentucky. Implementation of using different types of meshes with center pressure based method is feasible and straightforward. In order to increase flexibility in complex geometries, center pressure based method has been extended to use a variety of grid types, such as triangular, quadrilateral, tetrahedral, and hexahedral meshes. Mesh independent tests are also made to prove that current method can generate identical solutions for different mesh types. By using METIS for domain decomposition, excellent parallel load balance is achieved. In this paper, several test cases are presented—laminar incompressible flow past a flat plate, steady two/three-dimensional driven cavity flow, and unsteady two/three-dimensional flow over a circular cylinder for low Reynolds numbers. All these test cases yielded good agreements in comparison with previous computational or experimental results. A complete qualitative and quantitative study of two-dimensional flow over a circular cylinder for low Reynolds number is presented in this paper, which can be further used as a benchmark solution set for the development of new unsteady Navier-stokes solvers.

Acknowledgements

Support for this research was provided by Kentucky NASA EPSCoR, grant WKU 516140-02-06.

References

¹Karypis, G. and Kumar, V., A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices version 4.0. <http://www.cs.umn.edu/~karypis>, 1998.

²Rhie, C. M. and Chow, W. L., "Numerical study of the turbulent flow past an airfoil with trailing edge separation," *AIAA J.*, vol. 21, 1983, pp. 1525-1532.

³Ghia, U., Ghia, K. N., and Shin, C. T., "High-Resolution for Incompressible Flow Using the Navier-Stokes Equations and A Multigrid Method," *Journal of Computational Physics*, vol. 48, 1982, pp. 387-411.

⁴Ku, H. C., Hirsh, R. S. and Taylor, T. D., "A pseudospectral method for solution of the three-dimensional incompressible Navier-Stokes equations," *Journal of Computational Physics*, vol. 70, 1987, pp. 439-462.

⁵Shu, C., Wang, L., and Chew, Y. T., "Numerical Computation of Three-Dimensional Incompressible Navier-Stokes Equations in Primitive Variables Form by DQ method," *Int. J. Numer. Meth. Fluids*, vol. 43, 2003, pp. 345-368.

⁶Rogers, S. E. and Kwak, D., "Upwind differencing scheme for the time-accurate incompressible Navier-Stokes equations," *AIAA J.* Vol. 28, No. 2, 199, pp. 253-262.

⁷Ku, H. C., "Solutions of Flow in Complex Geometries by the Pseudospectral Element Method," *Journal of Computational Physics*, vol. 117, 1995, pp. 215-227.

⁸Alonso, J. J., Martinelli, L., and Jameson, A., "Multigrid Unsteady Navier-Stokes Calculations with Aeroelastic Applications," *AIAA paper 95-0048*.

⁹Liu, C., Zheng, X., and Sung, C. H., "Preconditioned Multigrid Methods for Unsteady Incompressible Flows," *Journal of Computational Physics*, vol. 139, 1998, pp. 35-57.

¹⁰Ronald, D. et al., "Transitioning Active Flow Control to Applications," *AIAA paper 99-3575*.

¹¹Qian, L. and vezza, M., "A Vorticity-Based Method for Incompressible Unsteady Viscous Flows," *Journal of Computational Physics*, vol. 172, 2001, pp. 515-542.

¹²Braza, M. and Persillon, H., "Prediction of Certain transition characteristics in the wake of a cylinder," *Proc. IUTAM Conf. Bluff Body Wake Instabilities*. Berlin: Springer-Verlag, 1992, pp. 279-328.

¹³Henderson, R. D., "Unstructured spectral element methods: parallel algorithms and simulations," PhD thesis. Princeton Univ, 1994.

¹⁴Roshko, A., "On the Development of Turbulent Wakes from Vortex Sheets," *NACA Report 1191*, 1954.

¹⁵Wille, R., "Karman Vortex Streets," *Advances in Applied Mechanics*, vol. 6, Academic, New York, 1960, pp. 273-287.

¹⁶Williamson, C. H. K., "Vortex dynamics in the cylinder wake," *Annu. Rev. Fluid Mech.* Vol. 28, 1996, pp. 477-539.

¹⁷Williamson, C. H. K., "Three-Dimensional Vortex Dynamics in Bluff Body Wakes," *Experimental Thermal and Fluid Science*, vol. 12, pp. 150-168, 1996.

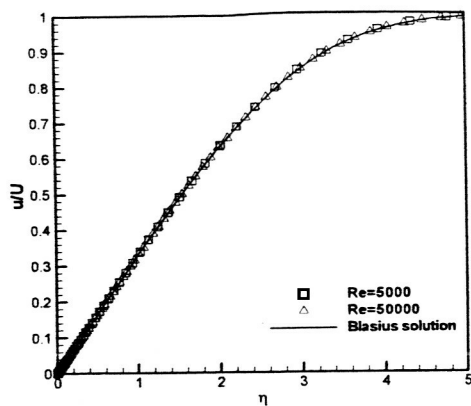


Fig. 5(a) Plot of η versus u/U at $x=4.5$ with present results and Blasius solution

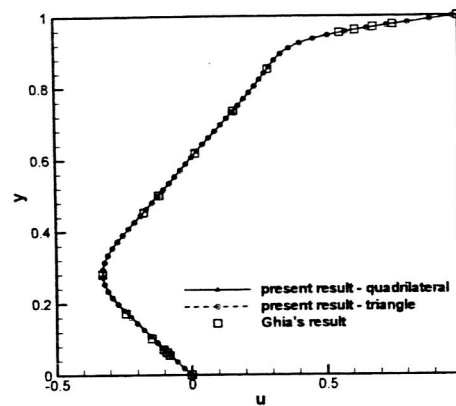


Fig. 7(a) The u -velocity profile along the horizontal center line for present results and Ghia's result.

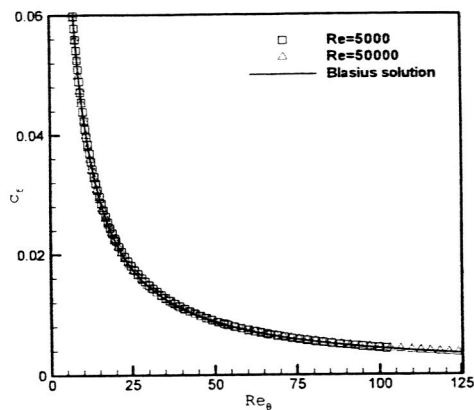


Fig. 5(b) The plot of Re_θ versus c_f with present results and Blasius solution

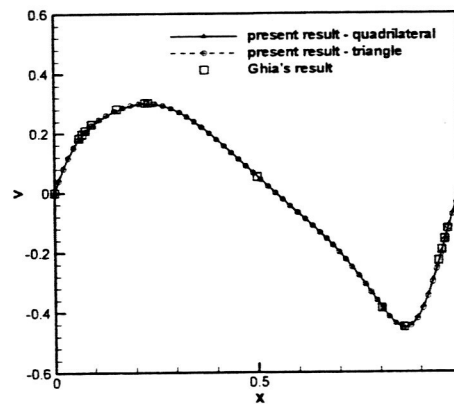


Fig. 7(b) The v -velocity profile along the vertical center line for present results and Ghia's result.

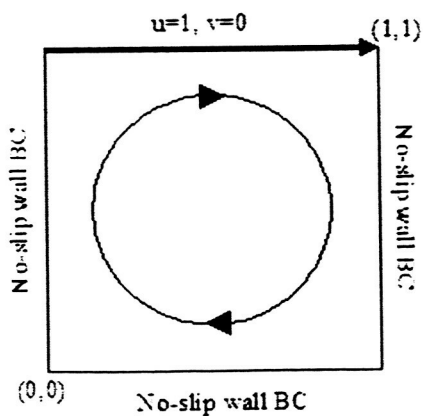


Fig. 6 The schematic diagram of two-dimensional driven cavity flow with boundary conditions

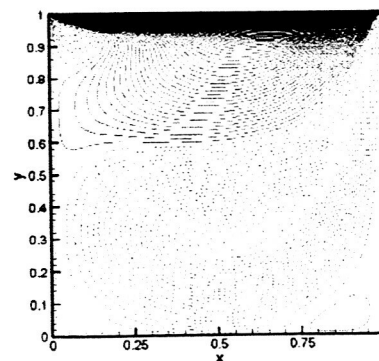


Fig. 8(a) The u -velocity contour plot for two-dimensional driven cavity flow at $Re=400$.

Table 1 Summary of present results and other computational and experimental results

Re	100	200	300	500	1000	1500
Present 2D results	$C_L=\pm 0.314$ $C_D=1.325\pm 0.008$ $S_r=0.165$	$C_L=\pm 0.642$ $C_D=1.318\pm 0.04$ $S_r=0.196$	$C_L=\pm 0.869$ $C_D=1.354\pm 0.072$ $S_r=0.21$	$C_L=\pm 1.115$ $C_D=1.406\pm 0.119$ $S_r=0.224$	$C_L=\pm 1.378$ $C_D=1.489\pm 0.198$ $S_r=0.239$	$C_L=\pm 1.553$ $C_D=1.575\pm 0.247$ $S_r=0.246$
Present 3D results	$C_L=\pm 0.322$ $C_D=1.327\pm 0.009$ $S_r=0.164$	$C_L=\pm 0.664$ $C_D=1.324\pm 0.042$ $S_r=0.195$				
Computational 2D results						
Rogers and Kwak [6] (5 th order)		$C_L=\pm 0.65$ $C_D=1.23\pm 0.05$ $S_r=0.185$				
Alonso et al. [7]				$C_L=\pm 1.046$ $C_D=1.217$ $S_r=0.224$		
Ku [8]	$C_L=\pm 0.228$ $C_D=1.33\sim 1.358$ $S_r=0.1675$			$C_L=\pm 1.03$ $C_D=1.212\sim 1.481$ $S_r=0.2203$	$C_L=\pm 1.242$ $C_D=1.187\sim 1.651$ $S_r=0.2326$	
Liu et al. [9]		$C_L=\pm 0.69$ $C_D=1.31\pm 0.049$ $S_r=0.192$				
Ronald et al. [10]			$C_L=\pm 0.841$ $C_D=1.34$ $S_r=0.2036$			
Qian and Vezza [11]					$C_D=1.52$ $S_r=0.24$	
Computational 3D results						
Braza and Persillon [12]			$S_r=0.202$			
Henderson [13]		$S_r=0.178$				
Experimental results						
Roshko [14]		$S_r=0.19$			$C_D=1.2$ $S_r=0.21$	
Wille [15]		$C_D=1.3$				
Williamson [16]	$S_r=0.166$		$S_r=0.203$			
Williamson [17]		$S_r=0.197$				